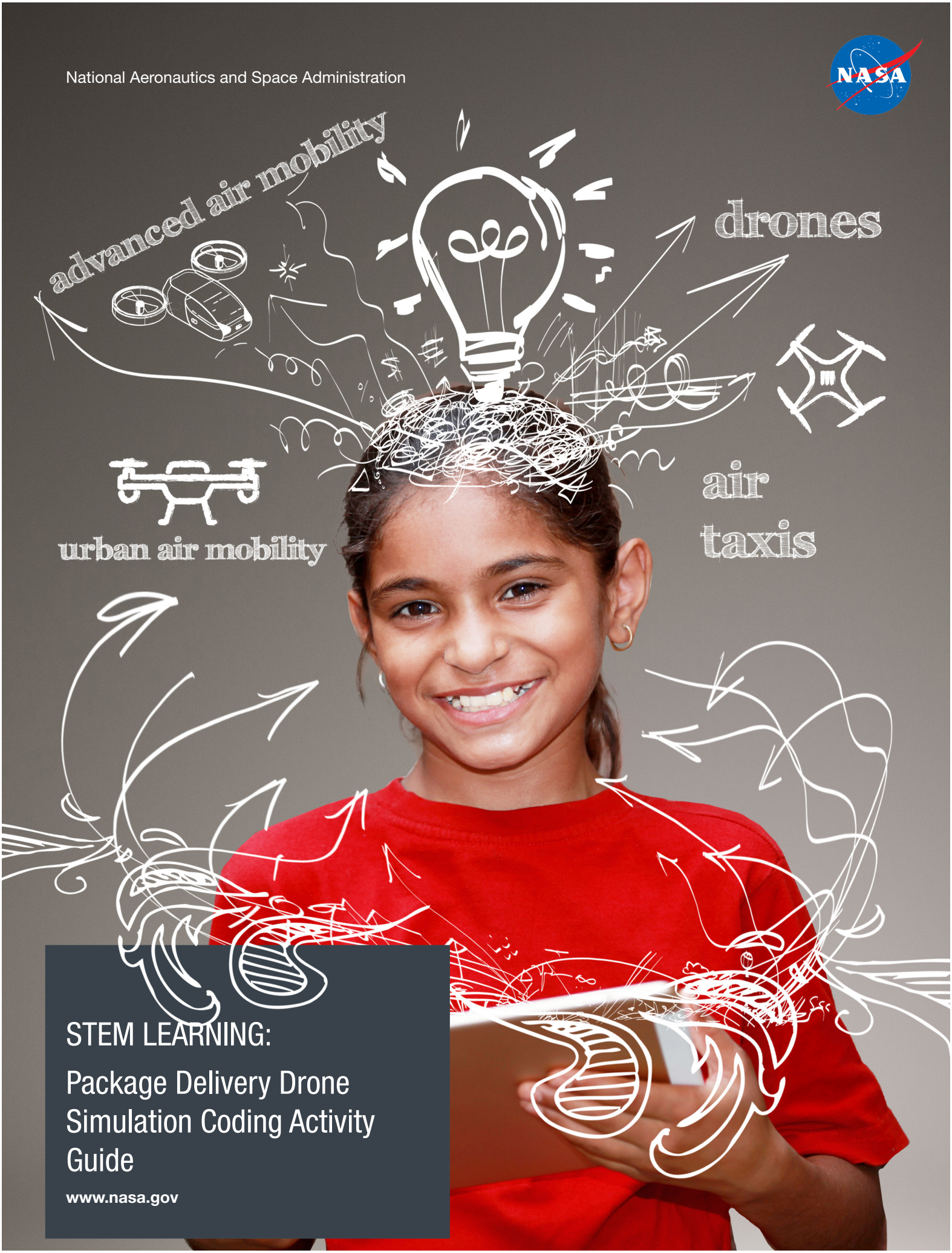
National Aeronautics and Space Administration



STEM LEARNING:

Package Delivery Drone Simulation Coding Activity Guide

www.nasa.gov

OVERVIEW

In this activity, students use Scratch, Snap!, or another programming language to create an interactive simulation of a drone navigating around a geofenced area to deliver a package. The simulation engages students in computational thinking, problem solving, and real-world application of mathematics.

Standards

Common Core State Standards for Mathematics

CCSS.6.RP.A.3: Use ratio and rate reasoning to solve real-world and mathematical problems, e.g., by reasoning about tables of equivalent ratios, tape diagrams, double number line diagrams, or equations.

CCSS.7.EE.B.4: Use variables to represent quantities in a real-world or mathematical problem, and construct simple equations and inequalities to solve problems by reasoning about the quantities.

CCSS.8.F.B.4: Use variables to represent quantities in a real-world or mathematical problem, and construct simple equations and inequalities to solve problems by reasoning about the quantities.

Next Generation Science Standards

Science and Engineering Practices

- Developing and using models
- Using mathematical and computational thinking

Crosscutting Concepts

- Scale, proportion, and quantity

Materials

- Computer or tablet with internet access (or download all content to run locally)
- Download file containing: Drone Sprites, Geofenced Area Sprites, Stages
- Programming language of your choice (Note: Snap! and Scratch can be downloaded to run locally). This guide gives instructions using Scratch.
  - Free Scratch account at: *http://scratch.mit.edu*
  - Free Snap! account at: *https://snap.berkeley.edu*

Grade Level

5th–12th

Suggested Time

1–2 hours

Management

1. This activity can be used to introduce students to coding or as a culminating activity after students have learned about the various coding blocks. The "Additional Resources" section contains other activities to practice block-based programming.

2. This activity guide includes recommendations for beginner and advanced-level programming. The advanced option requires more complex code, encourages higher-level thinking, and is intended for students with experience using block-based programming. By selecting different combinations of coding requirements, the activity can be modified based on the skill level of the students and the lesson's focus.

3. As students write their program, they should have it beta tested by another student. Beta testers should look for errors in the program's execution and provide feedback to the programmer.

4. If students complete the task with time remaining, they can be assigned additional higher-level requirements to keep them challenged. Additionally, they can come up with their own requirements and change their code to meet the new requirements.

5. Assessment:
   - Does the program meet all the assigned requirements?
   - Does the drone do what it is supposed to do with no errors?
   - Did the student go beyond the basics in creating their program?

Background Information

NASA is leading the nation to quickly open a new era in air travel called Advanced Air Mobility, or AAM. The vision of AAM is that of a safe, accessible, automated, and affordable air transportation system for passengers and cargo in both urban and rural locations.

According to recent NASA-commissioned market studies, by 2030 there will be as many as 500 million flights per year for package delivery services. The range of products that can be delivered by drones include food, medical supplies, retail purchases, and more.

To safely control aircraft in this new airspace, NASA has helped develop technology to create virtual barriers, or geofences, around areas where drones aren't allowed to fly. This is very important, because if drones fly in some areas, it can be dangerous for people and equipment. For example, drones flying too close to airports can interfere with the safe landing and taking off of the planes.

Block-Based Coding

Block-based coding simplifies the process of writing computer programs, making programming more accessible to students. In block-based coding, each block contains a component of the program and students combine blocks using a drag and drop editor.

One of the most widely used no cost block-based coding sites is Scratch (*http://scratch.mit.edu*), which the Massachusetts Institute of Technology manages. This site can be used online or downloaded and run locally. Snap! is an alternative coding site that is quite similar to Scratch. This guide uses Scratch for examples, instructions, etc.

There are many websites that provide guides, examples, and tutorials to help students and teachers learn how to code. Some useful websites include:

- *https://scratch.mit.edu/ideas*
- *https://en.scratch-wiki.info/wiki/Block-Based_Coding*
- *https://snap.berkeley.edu/index.html#examples*
- *https://snap.berkeley.edu/snapsource/help/SnapManual.pdf*

Programs in Scratch have three main parts: stages, sprites, and scripts. A stage is the background picture for the program, while sprites are other objects placed over the stage. The script(s) control the motion and interaction of the sprites.

Students should use the ball sprites that are included in Scratch as the start/end point, the delivery location, and the waypoints. The size of these sprites can be adjusted in the programming environment.

Waypoints are used to control the drone's flight by serving as an endpoint for a portion of the flightpath. More than one waypoint may be required to successfully maneuver around the geofenced area.

The download file contains:
Stages:

Stage 1

Stage 2

Stage 3

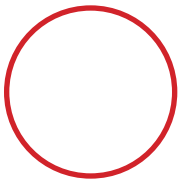
Students may be assigned or may choose any of these stages for their program. They must determine which area(s) within a stage to make off limits for drones; then, they must create a geofence, or virtual barrier, around that chosen area(s).

Sprites:

Drone 1

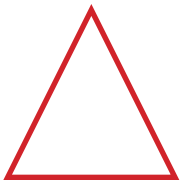
Drone 2

GeofenceCircle1



GeofenceCircle2

GeofenceTriangle1



GeofenceTriangle2

GeofenceRectangle1

GeofenceRectangle2

Students choose one of the drone sprites to move around in their program.

Students can use any combination of the geofence sprites to create the barrier around the area of the stage they choose to be off limits. Initially, the barrier should be yellow. If the drone flies into the area, the barrier should turn red.

Coding Exercise

For this activity, choose a stage to use for your program. On the stage, choose what area(s) to be off limits for the drone and create barriers, or geofences, around them. Pick a start/end point for the drone and a location where it will fly to drop off its package. The flightpath must take the drone completely around at least one geofenced area.

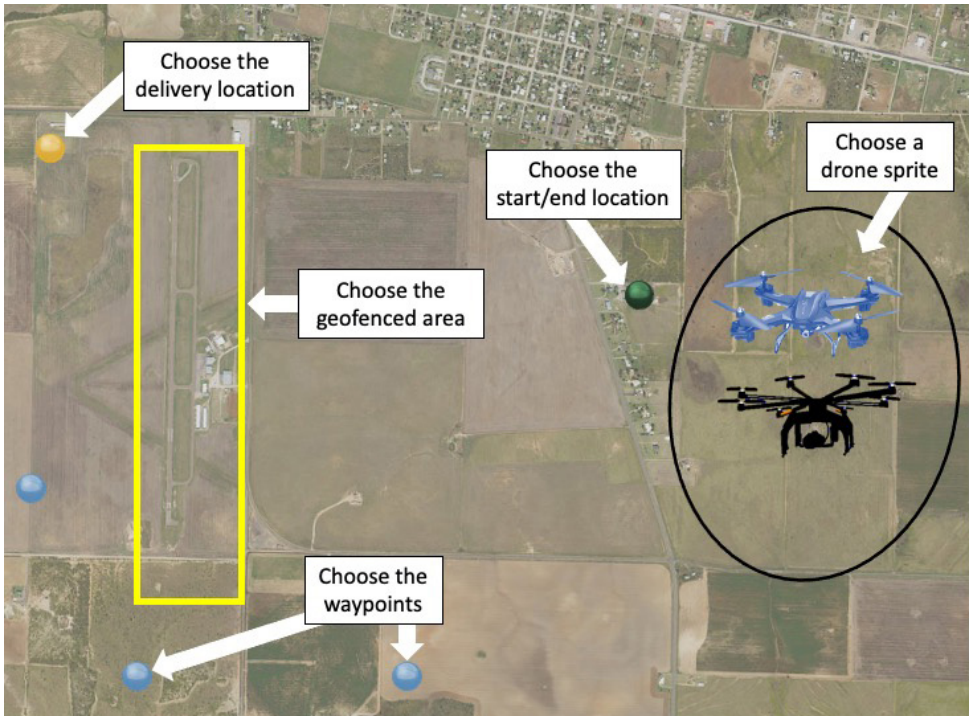
To create the drone's flightpath, add as many waypoints as needed. The drone will fly from the start point to the first waypoint and then on to the second waypoint, and so on. One of the waypoints should be the delivery location. The drone will end its flight back at the start/end point.

If the drone flies into a geofenced area, the geofence should turn from yellow to red. Also, the drone should immediately stop flying.

Coding Requirements

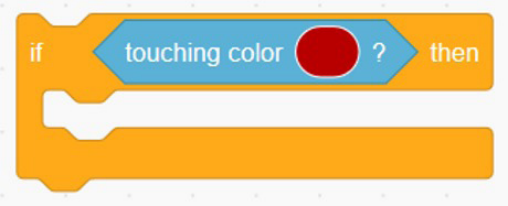
Beginner:

- ☐ Choose a stage (unless one is assigned)
- ☐ Choose a drone sprite
- ☐ Choose what area(s) to make off limits
- ☐ Add geofence sprite(s) to create a barrier around the off-limits area
- ☐ For each geofence sprite, add a costume to enable it to change color
- ☐ Choose the start/end point and add a sprite (the included ball sprite works well)
- ☐ Choose the delivery location and add a sprite (the included ball sprite works well; use a different color than the start/end point)
- ☐ Choose appropriate sizes for the sprites. The drone will not be to scale with the background, but this is necessary.
- ☐ Add additional waypoints as necessary to guide the drone around the geofenced area (the included ball sprite works well; use a different color than the other ball sprites)
- ☐ Write script so the drone moves from waypoint to waypoint to deliver the package and return to the start/end point
- ☐ Pause at the delivery location and alert the user that it is dropping off the package
- ☐ Flightpath takes the drone completely around at least one geofenced area
- ☐ Add script to turn geofence sprites red and stop the program if the drone enters a geofenced area
- ☐ Speed of the drone should be 100 pixels per second



Advanced options (choose which ones you want):

- ☐ Have the drone change size to simulate takeoffs and landings (it starts small, gets bigger for a takeoff, and gets smaller again for a landing)
- ☐ Add additional waypoint(s)
- ☐ Add manual controls so the user can control the drone's movement
- ☐ Add a button that the user can click for directions
- ☐ Instead of using the geofenced sprites, draw the geofences on the background stage and use the "touching color" block to detect if the drone is touching the geofenced area



Setting up the Programming Environment

Beginner:

1. Create a Scratch account:

   If working in the cloud, students need to create individual or team accounts at *http://scratch.mit.edu*. This is where they sign in and create a new project. Alternatively, Scratch can be downloaded and run locally on a computer. This way, students can work on projects without an active internet connection.

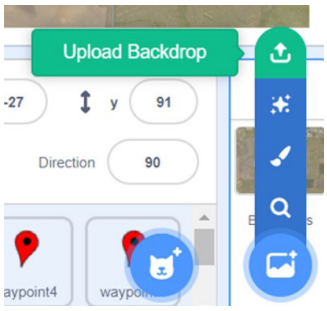
   If Snap! is preferred over Scratch as the programming environment, it can be accessed at *https://snap.berkeley.edu*. As with Scratch, students need to create an account on Snap! and it can be downloaded and run locally.

2. Download and unzip files:

   The zipped file contains the files needed for this activity. The contents of this zipped file are shown in the “Block-Based Coding” section of this guide.

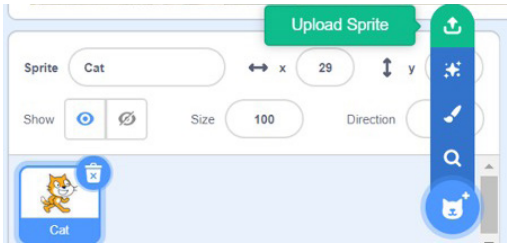
3. Select and add a stage:

   Upload one of the three stages found in the zipped file. The “Upload Backdrop” button is in the lower-right section of the Scratch programming environment.



4. Add sprites:

   Delete any sprites that are in the program by clicking on the garbage can button next to the sprite. Then, click the “Upload Sprite” button to add a drone sprite. Use the “Upload Sprite” button to also upload any yellow geofence sprites and as many waypoints as you need (use the ball sprites in Scratch for the waypoints).

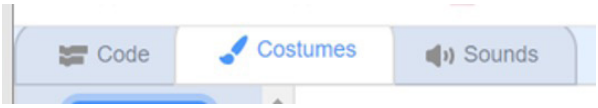


5. Edit sprites' sizes and/or direction:

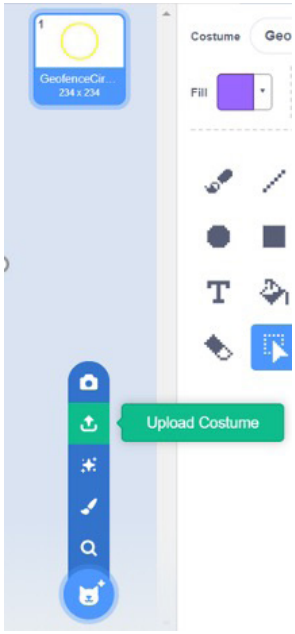
   Drag the sprites to where you want them on the stage. For each sprite, you can adjust the size to ensure all sprites are relatively proportional in size to one another. You can also click on the “Direction” box and rotate sprites. This may be necessary to align the geofence sprite with the area(s) you want to be off limits.

6. Add costume(s):

   Geofences will initially be yellow. If the drone flies into a geofenced area, the geofence sprite will need to turn red. To do this, you need to upload a costume.



   Once you select the sprite in the bottom right portion of the screen, you need to click the “Costume” tab. Then, click the “Upload Costume” button and add the corresponding red version of the geofence sprite.



   When writing scripts, you can change the costume at any time. So, when the script detects that the drone has run into the geofenced area, the geofence sprite is changed to red.

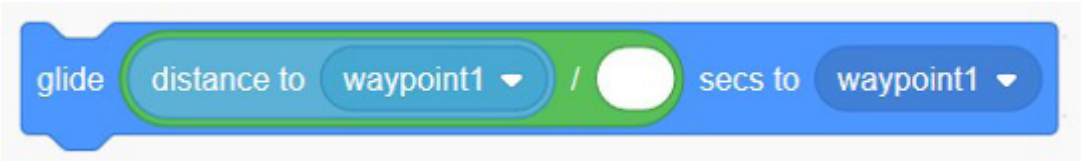
Coding Tips

1. Positioning and moving sprites:

   Locations on the stage are indicated using a set of coordinates similar to graphing points on a piece of graph paper. The origin has the coordinates (0, 0) and is located in the center of the stage.

   You can use the Scratch commands “go to...” and “glide to...” to move your drone sprite to the location of another sprite. This is where the waypoints, which are sprites, are useful—they allow you to move your sprite from point to point to determine the flightpath.

   To control the speed of the sprite, you can use:



   Scratch calculates the distance the drone has to move to a point (waypoint1 in this example). Dividing this distance by the desired speed gives the number of seconds needed and helps control the sprite’s speed.

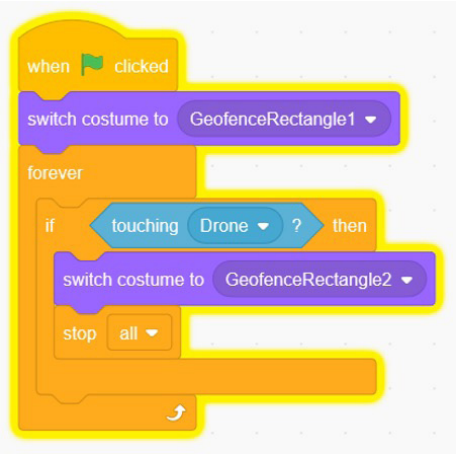
2. Using costumes:

   Costumes in Scratch are alternate forms of sprites. For example, if you have a yellow box that represents the border around a geofenced area, you can replace it with a red box to show that the drone has hit the geofence. To do this, you need to first upload whichever shape geofence sprite you need. Then, select that sprite and click the "Costumes" tab in the upper left portion of the programming environment. That is where you upload the alternate version. It is important that both costumes are the same shape.

3. Detecting collisions between sprites:

   An important aspect of this activity is detecting when the drone sprite makes contact with the geofence sprite. Scratch has commands that help with this. To detect a collision, you need to add a script for the geofence sprite.

   This script should begin when your program does. First, it changes the geofence costume to the yellow outline. Then it continuously checks to see if the drone sprite is touching the geofence sprite. If so, it changes the costume so the geofence is red. It also stops the program at that point so the drone cannot enter the off-limits area.



Extension Ideas

There are many ways this activity can be extended for students that quickly grasp the concept.

- Add a splash screen at the start of the program that gives the basic instructions.
- Create a new sprite for the drone using Scratch's paint tool or other software.
- Create two layers of geofencing—the drone slows down if it reaches the first layer and stops if it reaches the second layer.
- Add sound effects (there are sound effects built into Scratch or students can record their own).
- Create a new background (stage) using mapping software. This could include the area around where the student lives.
- Program determines and records whether the drone delivered the package to the correct delivery location (i.e., did it touch the delivery location sprite during the drop off?).
- Add multiple delivery locations.

Grading Rubric

Rubric Category	Score
Program Execution and Output • Program simulates a drone flying to deliver a package. • Program meets the coding requirements (mark the requirements met): ☐ Stage and one drone sprite added ☐ Geofence present with appropriate sizing and positioning ☐ Start/end point and delivery location added and are visually dissimilar ☐ Adequate number of waypoints added and they are visually dissimilar from start/ end point and delivery location ☐ Sprites' sizes make a visually appealing program ☐ Drone moves between waypoints ☐ Drone pauses at the delivery location and indicates that it is delivering the package ☐ Flightpath takes the drone completely around at least one geofenced area ☐ Geofenced areas turn red if contacted by the drone sprite ☐ Speed of the drone is 100 pixels per second	_____/4
Program Execution and Output • All errors and bugs are eliminated. • Program is well organized and code runs efficiently. • Program goes beyond the minimum requirements.	_____/4
Project Management • Project completed on time. • Time given is used productively to work on project. • Programmer had others beta test the program to help find errors.	_____/4
TOTAL	_____/12

4 (Advanced) = All criteria are met and followed with very few mistakes
3 (Proficient) = Most criteria are met with few mistakes
2 (Developing) = Many criteria are not met and/or there are many mistakes
1 (Beginner) = Most criteria are not met
0 (Very Little to No Effort) = Very little to no effort was put forth to meet criteria

Additional NASA Resources for Practicing Block-Based Coding

- Attack of the Drones:
  - https://www.nasa.gov/sites/default/files/atoms/files/aam-attack-of-the-drones-coding-activity-guide.pdf
  - NASA's Aeronautics Research Mission Directorate developed this tutorial-style activity to guide students through the creation of a side-scrolling game which highlights some of the aspects of Advanced Air Mobility (AAM).
- Explore Mars with Scratch:
  - *https://www.jpl.nasa.gov/edu/teach/activity/explore-mars-with-scratch/*
  - NASA's Jet Propulsion Lab designed this activity, which takes a student through the steps of creating a simple game where the user controls a rover on Mars.
- Crew Orbital Docking Simulation:
  - *https://www.nasa.gov/stem-ed-resources/crew-orbital-docking-simulation-coding-sim.html*
  - Designed as part of the Commercial Crew Program, this NASA activity challenges students to create a simulation of a commercial spacecraft docking with the International Space Station.

National Aeronautics and Space Administration

Headquarters
300 E Street SW
Washington, DC 20546

www.nasa.gov

EP-2020-04-522-HQ